



Chapter 1

Sat-based & Bounded model checking

Course "Specification and Verification of Parallel Systems"

Martin Steffen, Paul Kobialka

Autumn 2025



Section

Introduction

Autumn 2025

Advantages of MC

- no proofs, “push button”¹
- diagnostic **counterexamples**
- logics used for MC can express many concurrency problems

¹One could debate that.



IN5110 –
Specification and
Verification of
Parallel Systems

Introduction

SAT solving and
SMT

Reducing bounded
model checking to
SAT

Main “disadvantage”



IN5110 –
Specification and
Verification of
Parallel Systems

Introduction

SAT solving and
SMT

Reducing bounded
model checking to
SAT

- **state space explosion problem** (aka state explosion problem)
- problem “solution” space grows *exponential* is the problem “description” space
 - notably reachable state space exponential in the number of processes

The 4 big breakthroughs combatting the SSEP

Apart from

- advances in data structures,
- software engineering,
- tricks, optimizations, heuristics and
- general advances in processing power/memory.

Clarke identifies the following

“big 4” breakthroughs

1. **symbolic** techniques (notably using BDDs)²
2. **partial order reduction**
3. **bounded model checking**
4. CEGAR, localisation reduction

²See later presentations



IN5110 –
Specification and
Verification of
Parallel Systems

Introduction

SAT solving and
SMT

Reducing bounded
model checking to
SAT



Section

SAT solving and SMT

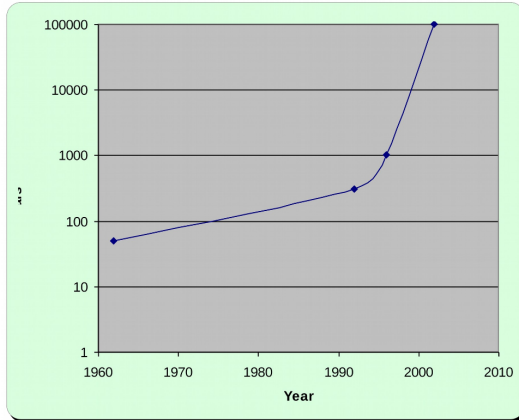
Autumn 2025



- (boolean) satisfiability
- famous, prototypical **NP-complete** problem

SAT solver progress

- highly competitive field
- yearly “SAT-competition”³



taken from [4]

³<http://www.satcompetition.org/>



IN5110 –
Specification and
Verification of
Parallel Systems

Introduction

SAT solving and
SMT

Reducing bounded
model checking to
SAT

Bounded model checking



IN5110 –
Specification and
Verification of
Parallel Systems

BMC starting point

Leverage sat-solving, a powerful a successful technique, to do model checking

Introduction

SAT solving and
SMT

Reducing bounded
model checking to
SAT

Cf.: Symbolic model checking and BDDs



IN5110 –
Specification and
Verification of
Parallel Systems

Introduction

SAT solving and
SMT

Reducing bounded
model checking to
SAT

- See separate presentation
- successful technique
- used (most prominently for HW) in industrial uses of MC
- Two ingredients of SMC
 - operating **symbolically** on representation of **sets** of states
 - use *BDDs* (= specific kind of graph representation of *boolean* functions) to represent and operate on them
- like SMC/BDD-based MC: **BMC** based on “boolean encodings”

Bad news: the MC problem/reachability is *not* a SAT problem :-)

- MC *here*:⁴
 - models are kind of transition systems/Kripke structures ...
 - spec's are “temporal logic” formulas

solving an MC problem

It all boils down to some form of fancy graph **reachability**

- “reachability”, however:
 - a form of “**fixpoint**” calculation⁵
 - fixpoints are emphatically **not** part of boolean logic.⁶

⁴The term “model checking”, i.e., solving $M \models^? \varphi$ can be applied in different settings as well. A boolean assignment can be seen as *model* of a propositional formula, for instance. That *is* of course a SAT problem. But we are interested transition systems satisfying a TL formula.

⁵see also the presentation about μ -calculus.

⁶They are not even part of first-order logic. Implicitly they are part in temporal logics, though (eventually, until etc.)



Good news: *bounded* MC can be seen as SAT :-)



IN5110 –
Specification and
Verification of
Parallel Systems

Introduction

SAT solving and
SMT

Reducing bounded
model checking to
SAT

less ambitious goal

Can I find an error (counterexample) in the behavior of the system considering **up-to k steps** from the initial states

- price to pay: no more “verification”⁷
- bug-hunting
- simple core idea

⁷but MC is typically verification of a model/abstraction anyhow and/or verification up until the MC runs out of time/memory.

LTL and “existential” LTL

- remember: LTL (linear time temporal logic) and definition of

$$S \models \varphi$$

- φ must hold for **all** paths of S
- If $S \not\models \varphi$ (error), then **exists** a paths π such that $\pi \not\models \varphi$

For explicitness' sake

path quantifiers⁸

$$\forall \varphi \quad \text{and} \quad \exists \varphi$$

- assume NNF

⁸one single quantifier as prefix to an LTL formula.



IN5110 –
Specification and
Verification of
Parallel Systems

Introduction

SAT solving and
SMT

Reducing bounded
model checking to
SAT

Terminology: witnesses

counterexample for

$$S \models \Box p \quad \text{corresponds to} \quad S \models \forall \Box p$$

corresponds to the question if there **exists** a **witness**⁹

$$\Diamond \neg p$$

- Goal: find **finite** (fixed bound) prefixes as **witness** to an **existential** model checking problem (LTL)
- conceptually easy if original $\forall \varphi$ is a safety prop.
- liveness? witness for $\exists \Box$?

⁹in logics in general, a witness is a thing (here a path) that gives (constructive) evidence to an existential formula



Terminology: witnesses

counterexample for

$$S \models \Box p \quad \text{corresponds to} \quad S \models \forall \Box p$$

corresponds to the question if there **exists** a **witness**⁹

$$\Diamond \neg p$$

- Goal: find **finite** (fixed bound) prefixes as **witness** to an **existential** model checking problem (LTL)
- conceptually easy if original $\forall \varphi$ is a safety prop.
- liveness? witness for $\exists \Box ? \Rightarrow$ loops

⁹in logics in general, a witness is a thing (here a path) that gives (constructive) evidence to an existential formula



Paths with and without loops



IN5110 –
Specification and
Verification of
Parallel Systems

Introduction

SAT solving and
SMT

Reducing bounded
model checking to
SAT

No loop



- only prefix with **back loop** can be witness for $\Box p$

(k, l) -loop



Given: TS/Kripke-structure. transition relation $\rightarrow$.

Definition

Assume $l \leq k$. A path π is a **(k, l) -loop** if $\pi_k \rightarrow \pi_l$ and

$$\pi = u \cdot v^\omega \quad (1)$$

with

$$u = \pi_0 \dots \pi_{l-1} \quad \text{and} \quad v = \pi_l \dots \pi_k$$

A path π is a **k -loop** if there exists an l with $0 \leq l \leq k$ s.t. π is a (k, l) -loop

- remember: paths π are (infinite) sequences of “states” (worlds)
- loops here is about those states (not “edges” of the picture)

Bounded semantics

- remember the “normal” semantics of LTL from before, relating formulas and paths
- $\llbracket \varphi \rrbracket$ or $\pi \models \varphi$
- now: the new “looping paths” (k -loops) as basis for **bounded semantics**, i.e., basis for BMC
- note: “finite” prefixes (loops) can give information for infinite paths, thus serve as witnesses
- boundes semantics for path
 - with loop:** “unchanged”
 - without loop:** be aware of the **cut-off** and be **pessimistic**



IN5110 –
Specification and
Verification of
Parallel Systems

Introduction

SAT solving and
SMT

Reducing bounded
model checking to
SAT

Bounded semantics: for loops



IN5110 –
Specification and
Verification of
Parallel Systems

Definition (Bounded semantics: with lasso)

Let π be a k -loop. A formula φ is *valid* along π *with bound* k , written

$$\pi \models_k \varphi \quad \text{iff} \quad \pi \models \varphi .$$

Introduction

SAT solving and
SMT

Reducing bounded
model checking to
SAT

Bounded semantics: without loops

Definition

Let π be a path which is *not* a k -loop. Then an LTL formula φ is *valid along π with bound k* , written

$$\pi \models_k \varphi ,$$

iff $\pi \models_k^0 \varphi$, given below.

- earlier $\pi \models \varphi$, corresponding here to $\models^0$
- k is treated as “cut-off”:
- what comes *afterward*: **unknown**
- if **in doubt**: “false”, i.e., the path is not valid/does not satisfy the formula in the bounded manner
 - for $\bigcirc$: don’t “look” beyond k
 - for $\square$: be pessimistic
 - for $\diamond$: positive answer at least possible within the bound



IN5110 –
Specification and
Verification of
Parallel Systems

Introduction

SAT solving and
SMT

Reducing bounded
model checking to
SAT

Bounded semantics: without loops ($\models_k^i$)

Definition (Bounded semantics: without lasso)

$$\begin{array}{ll} \pi \models_k^i p & \text{iff } p \in L(\pi_i) \\ \pi \models_k^i \neg p & \text{iff } p \notin L(\pi_i) \\ \pi \models_k^i \varphi_1 \wedge \varphi_2 & \text{iff } \pi \models_k^i \varphi_1 \text{ and } \pi \models_k^i \varphi_2 \\ \pi \models_k^i \varphi_1 \vee \varphi_2 & \text{iff } \pi \models_k^i \varphi_1 \text{ or } \pi \models_k^i \varphi_2 \end{array}$$

$$\begin{array}{ll} \pi \models_k^i \Box \varphi & \text{is always false} \\ \pi \models_k^i \Diamond \varphi & \text{iff } \exists j. i \leq j \leq k. \pi \models_k^j \varphi \\ \pi \models_k^i \bigcirc \varphi & \text{iff } i < k \text{ and } \pi \models_k^{i+1} \varphi \\ \pi \models_k^i \varphi_1 U \varphi_2 & \text{iff } \exists j. i \leq j \leq k. \pi \models_k^j \varphi_2 \text{ and } \forall n, i \leq n < j. \pi \models_k^n \varphi_1 \\ \pi \models_k^i \varphi_1 R \varphi_2 & \text{iff } \exists j. i \leq j \leq k. \pi \models_k^j \varphi_1 \text{ and } \forall n, i \leq n < j. \pi \models_k^n \varphi_2 \end{array}$$



IN5110 –
Specification and
Verification of
Parallel Systems

Introduction

SAT solving and
SMT

Reducing bounded
model checking to
SAT

Bounded $\rightarrow$ unbounded semantics

- Note, the connection is done for **existential** LTL (formulas of the form $\exists\varphi$, not like $\forall\varphi$)
- unbounded semantics as **limit** of the bounded ones (for all/arbitrary bounds k)

Lemma (Easy direction (per path))

$$\pi \models_k \varphi \text{ implies } \pi \models \varphi$$

Lemma (For TSs/KSs)

$$S \models \exists\varphi \text{ implies } S \models_k \exists\varphi \text{ for some } k \geq 0$$

Theorem

$$S \models \exists\varphi \text{ iff } S \models_k \exists\varphi \text{ for some } k \geq 0$$





Section

Reducing bounded model checking to SAT

BMC via SAT

- so far:
 - definition of the bounded MC **problem**
 - we convinced ourself: BMC approximates MC (at least for existential path formulas)
- Now: reduce to sat-solving

Goal

$\llbracket S, \varphi \rrbracket_k$ is *satisfiable* iff π is a witness for φ

- **sat**-problems: formula with (propositional) variables
- encoding given in 3 parts. given k
 1. valid initial path for S and
 2. satisfaction of formula if
 - there's a loop or
 - there's no loop
- remember symbolic CTL model checking



IN5110 –
Specification and
Verification of
Parallel Systems

Introduction

SAT solving and
SMT

Reducing bounded
model checking to
SAT

Definition (Transition system)

A **transition system** (or **Kripke structure**) is a tuple $(S, I, \rightarrow, V)$ where S is the set of states, $I \subseteq S$ the set of initial states, $\rightarrow \subseteq S \times S$ the transition relation, and $V : S \rightarrow 2^{AP}$ the *valuation* function (aka. (state) labelling function).

- transition relation: a predicate:¹⁰ $\rightarrow : S^2 \rightarrow \mathbb{B}$
- initial states: a predicate $I : S \rightarrow \mathbb{B}$

¹⁰others write $T(s_1, s_2)$ for our infix relational notation $s_1 \rightarrow s_2$, where T is the transition relation predicate.

1st component: Translating S



IN5110 –
Specification and
Verification of
Parallel Systems

Introduction

SAT solving and
SMT

Reducing bounded
model checking to
SAT

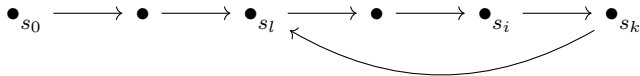
- remember transition system/Kripke structures S
 - states s_i . Consider s_i as *variables*
 - transition relation: as predicate $T(s_k, s_l)$, we write still infix $s_k \rightarrow s_l$
- **unfolding** of the transition relation

$$\llbracket T \rrbracket_k \triangleq I(s_0) \wedge \bigwedge_{i=0}^{k-1} s_i \rightarrow s_{i+1} \quad (2)$$

- remember in CTL how we encoded $S \times S$
- states in KS: *propositional variables* s_k

Loop condition

- Remember the def. of (k, l) -loop



- simple abbreviation

$${}_l L_k \triangleq s_k \rightarrow s_l$$

- loop condition** holds¹¹ iff there is a **back** loop from a state s_k back to a previous state s_l (which can be s_k)

Definition (Loop condition)

$$L_k \triangleq \bigvee_{l=0}^k {}_l L_k$$

¹¹resp. it will hold when applied to a path consisting of a sequence of states s_i , which are considered as propositional variables, as said. the word “back” makes sense only if one interprets the variables to be “in a sequence”.



Successor in a loop

a rather unsurprising definition: define “successor”

$\text{succ}(i)$ of i in a (k, l) -loop as

- $\text{succ}(i) = i + 1$ for $i < k$
- $\text{succ}(i) = l$ for k



IN5110 –
Specification and
Verification of
Parallel Systems

Introduction

SAT solving and
SMT

Reducing bounded
model checking to
SAT

2nd component: translating formula with a loop



IN5110 –
Specification and
Verification of
Parallel Systems

propositional part: boring

$$\begin{aligned}l\llbracket p \rrbracket_k^i &\triangleq p(s_i) \\l\llbracket \neg p \rrbracket_k^i &\triangleq \neg p(s_i) \\l\llbracket \varphi_1 \wedge \varphi_2 \rrbracket_k^i &\triangleq l\llbracket \varphi_1 \rrbracket_k^i \wedge l\llbracket \varphi_2 \rrbracket_k^i \\l\llbracket \varphi_1 \vee \varphi_2 \rrbracket_k^i &\triangleq l\llbracket \varphi_1 \rrbracket_k^i \vee l\llbracket \varphi_2 \rrbracket_k^i\end{aligned}$$

Introduction

SAT solving and
SMT

Reducing bounded
model checking to
SAT

Cont'd

Actually straightforward

- loop $\rightarrow$ no cut-off $\rightarrow$ “standard semantics”
- remember *unrolling* of fixpoints¹²

temporal part: a bit more interesting

$$l[\Box\varphi]_k^i \triangleq l[\varphi]_k^i \wedge l[\Box\varphi]_k^{succ(i)}$$

$$l[\Diamond\varphi]_k^i \triangleq l[\varphi]_k^i \vee l[\Diamond\varphi]_k^{succ(i)}$$

$$l[\bigcirc\varphi]_k^i \triangleq l[\varphi]_k^{succ(i)}$$

$$l[\varphi_1 \ U \ \varphi_2]_k^i \triangleq l[\varphi_1]_k^i \vee l[\varphi_1 \ U \ \varphi_2]_k^{succ(i)}$$

$$l[\varphi_1 \ R \ \varphi_2]_k^i \triangleq l[\varphi_2]_k^i \wedge l[\varphi_1 \ R \ \varphi_2]_k^{succ(i)}$$

¹²Cf. also the presentation about the μ -calculus. Also in the construction of the Büchi-automaton from an LTL formula, that unrolling played a role (for U).



IN5110 –
Specification and
Verification of
Parallel Systems

Introduction

SAT solving and
SMT

Reducing bounded
model checking to
SAT

Translation without a loop

- same principles
- “index” l not needed
- instead of the more complex $\text{succ}(i)$: simply $i + 1$.
- otherwise: the definition stays “the same”)



IN5110 –
Specification and
Verification of
Parallel Systems

Introduction

SAT solving and
SMT

Reducing bounded
model checking to
SAT

3rd component: translating formula without a loop

Inductive case $\forall i \leq k$:

propositional part: boring again

$$\begin{aligned} \llbracket p \rrbracket_k^i &\triangleq p(s_i) \\ \llbracket \neg p \rrbracket_k^i &\triangleq \neg p(s_i) \\ \llbracket \varphi_1 \wedge \varphi_2 \rrbracket_k^i &\triangleq \llbracket \varphi_1 \rrbracket_k^i \wedge \llbracket \varphi_2 \rrbracket_k^i \\ \llbracket \varphi_1 \vee \varphi_2 \rrbracket_k^i &\triangleq \llbracket \varphi_1 \rrbracket_k^i \vee \llbracket \varphi_2 \rrbracket_k^i \end{aligned}$$



IN5110 –
Specification and
Verification of
Parallel Systems

Introduction

SAT solving and
SMT

Reducing bounded
model checking to
SAT

Non-loop-case (cont'd)

Inductive case $\forall i \leq k$:

temporal part: a bit more interesting

$$\llbracket \Box \varphi \rrbracket_k^i \triangleq \llbracket \varphi \rrbracket_k^i \wedge \llbracket \Box \varphi \rrbracket_k^{i+1}$$

$$\llbracket \Diamond \varphi \rrbracket_k^i \triangleq \llbracket \varphi \rrbracket_k^i \vee \llbracket \Diamond \varphi \rrbracket_k^{i+1}$$

$$\llbracket \bigcirc \varphi \rrbracket_k^i \triangleq \llbracket \varphi \rrbracket_k^{i+1}$$

$$\llbracket \varphi_1 \ U \ \varphi_2 \rrbracket_k^i \triangleq \llbracket \varphi_1 \rrbracket_k^i \vee \llbracket \varphi_1 \ U \ \varphi_2 \rrbracket_k^{i+1}$$

$$\llbracket \varphi_1 \ R \ \varphi_2 \rrbracket_k^i \triangleq \llbracket \varphi_2 \rrbracket_k^i \wedge \llbracket \varphi_1 \ R \ \varphi_2 \rrbracket_k^{i+1}$$

- base case: $\llbracket \varphi \rrbracket_k^{k+1} \triangleq false$



IN5110 –
Specification and
Verification of
Parallel Systems

Introduction

SAT solving and
SMT

Reducing bounded
model checking to
SAT

Putting it together



IN5110 –
Specification and
Verification of
Parallel Systems

Introduction

SAT solving and
SMT

Reducing bounded
model checking to
SAT

$$\begin{aligned} \llbracket S, \varphi \rrbracket_k \triangleq & \llbracket S \rrbracket_k \wedge \\ & ((\neg L_k \wedge \llbracket \varphi \rrbracket_k^0) \\ & \vee (\bigvee_{l=0}^k (l L_k \wedge l \llbracket \varphi \rrbracket_k^0))) \end{aligned} \quad (3)$$

Theorem

$$\llbracket S, \varphi \rrbracket_k \text{ satisfiable } \text{ iff } S \models_k \exists \varphi .$$

Further info

- The technical slides here recap parts of the journal article [1] by the inventors of BMC
- BMC for software [2]
- Survey [5]



IN5110 –
Specification and
Verification of
Parallel Systems

Introduction

**SAT solving and
SMT**

Reducing bounded
model checking to
SAT

Bibliography

- [1] Biere, A. (2021). Bounded model checking. In *Handbook of satisfiability*, pages 739–764. IOS press.
- [2] Clarke, E., Kroening, D., and Lerda, F. (2004). A tool for checking ansi-c programs. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 168–176. Springer.
- [3] Clarke, E. M. (2008). Model checking – my 27-year quest to overcome the state explosion problem. In Cervesato, I., Veith, H., and Voronkov, A., editors, *Logic for Programming, Artificial Intelligence, and Reasoning: 15th International Conference, LPAR 2008, Doha, Qatar, November 22–27, 2008. Proceedings*, Lecture Notes in Artificial Intelligence, pages 182–182. Springer Verlag.
- [4] Clarke, E. M. (2017). SAT-based bounded and unbounded model checking. Available electronically on the net. Data of publication unknown.
- [5] Prasad, M. R., Biere, A., and Gupta, A. (2005). A survey of recent advances in sat-based formal verification. *International Journal on Software Tools for Technology Transfer*, 7(2):156–173.

Introduction

SAT solving and SMT

Reducing bounded model checking to SAT